Yale University
Department of Computer Science

A Model of Onion Routing with Provable Anonymity

Aaron Johnson

YALEU/DCS/TR-1368
August 30, 2006

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
30 AUG 2006		**00-08-2006 to 00-08-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Model of Onion Routing with Provable Anonymity	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Yale University,Department of Computer Science,PO Box 208285,New Haven,CT,06520-8285**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**19**	

A Model of Onion Routing with Provable Anonymity

Aaron Johnson*
Department of Computer Science
Yale University
New Haven, CT
ajohnson@cs.yale.edu

Abstract

Onion routing is a scheme for anonymous communication that is designed for practical use. It has not been modeled formally, however, and therefore its anonymity guarantees have not been rigorously analyzed. We give an IO-automata model of an onion-routing protocol and, under possibilistic definitions, characterize the situations in which anonymity and unlinkability are guaranteed.

1 Introduction

Anonymity networks allow users to communicate while hiding their identities from one another and from third parties. We would like to design such networks with strong anonymity guarantees but without incurring high communication overhead or much added latency. Designs have been proposed [4, 5, 9, 18, 13, 6, 10, 1] that meet these goals to varying degrees.

Onion routing [11, 17, 20] is a practical anonymity-network scheme with relatively low overhead and latency. Several implementations have been made [17, 20], and it was even a basis for a commercial system [2]. A recent iteration of the basic design is the Tor system [9]. Tor has been implemented and, as of June 2006, consists of over 600 routers, each processing an average of over 7GB of traffic a week.

However, because onion routing is a practical, rather than theoretical, design, rigorous guarantees of the anonymity it provides have not been made. To this end, we propose a formal model of onion routing, based on the connection-oriented Tor protocol, using IO automata. We then suggest possibilistic definitions of anonymity and unlinkability within this model and provide necessary and sufficient conditions for them to be provided to a user.

2 Analyzing Onion Routing

An onion-routing network consists of a set of onion routers. To send data, a client chooses a sequence of routers, called a *circuit*, through which the data will be routed. Each packet is encrypted multiple times before sending, once for each router in the circuit and in reverse order of the routers' appearance in the circuit. This layered structure is called an *onion*. Encryption can be done with a private key that is shared with each router or with the public key of each router. Each router uses its key to decrypt the onion as it is forwarded through the circuit. The onion structure helps the sender to hide the data contents from all but the last router in the circuit, and, because the decryption changes the data representation, the onion also makes it harder for a network observer to follow the path the data takes through the network.

There has been work done to analyze the effectiveness of onion routing. Syverson et al. [21] consider a system, similar to Tor, under various configurations and adversary models. They examine the probability

*Supported by NSF grant number 0428422 and ONR grant number N00014-01-1-0795.

that certain types of anonymity compromises occur, but leave open whether other types are still possible. Camenisch and Lysyanskaya [3] give a cryptographic definition of a connectionless version of onion routing and prove that the cryptography it uses doesn't leak any information to the onion routers other than the previous and next routers. Mauw et al. [15] do an analysis of connectionless onion routing that is very similar to the analysis done in this paper. Their model is expressed in a process algebra. They use a possibilistic definition of anonymity, and show that onion routing provides sender and receiver anonymity against a passive global adversary.

Our approach is to formalize an onion-routing protocol with a network of automata. The protocol is based on Tor. We consider the case of an adversary controlling a fixed set of routers that are allowed to run any arbitrary automata. Then we show that this adversary cannot distinguish among certain user circuit configurations. In particular, the adversary cannot determine which user owns a circuit unless the adversary controls the first hop. The set of users which have an uncompromised first hop form a sender "anonymity set," among which the adversary cannot distinguish. Similarly, the adversary cannot determine the last router of a circuit unless it controls it or the penultimate router. Such circuits provide receiver anonymity. These two results justify considering only those cases considered in [21]. Also, a user is "unlinkable" to his destination when he has receiver anonymity or his sender anonymity set includes another sender with a destination that is different or unknown to the adversary.

We only consider possibilistic anonymity here. An action by user u is considered to be anonymous when there exists some system in which u doesn't perform the action, and that system has an execution that is consistent with what the adversary sees. The actions for which we consider providing anonymity are sending messages, receiving messages, and communicating with a specific destination. A more formal treatment of this anonymity concept is given by Halpern and O'Neill [12]. We do not consider our system probabilistically, as is required in more refined definitions of anonymity [19, 7]. It is possible to examine probabilistic definitions within our system by defining a probability measure over executions, as discussed in [12], or over initial states.

3 Model

3.1 Distributed system

Our model of onion routing is based on IO automata [14]. This formalism allows us to express an onion-routing protocol, model the network, and make precise the adversary's capabilities. One of its benefits is that it models asynchronous computation and communication. Another is that every action is performed by a single agent, so the perspective of the adversary is fairly clear.

Model onion routing as a fully connected asynchronous network of IO automata. The network is composed of FIFO channels. There is a set of users U and a set of routers R. Let $N = U \cup R$. We use the term *agent* to refer to an element of N. It is possible that $U \cap R \neq \emptyset$. In this case, user and router automata exist on the same processor. We assume that the users all create circuits of a fixed length l.

Each router-and-user pair shares a set of secret keys; however, the router does not know which of its keys belong to which user. We assume that all keys in the system are distinct. This separates, for now, key distribution from the rest of the protocol. Let K be the keyspace.

Let P be the set of control messages, and $\bar{P}$ be the extension of P by encryption with up to l keys. The control messages will be tagged with a link identifier and circuit identifier when sent, so let the protocol message space be $M = \mathbf{N}_+ \times \mathbf{N}_+ \times \bar{P}$. We denote the encryption of $p \in P$ using key k with $\{p\}_k$, and the decryption with $\{p\}_{-k}$. For brevity, the multiply encrypted message $\{\{p\}_{k_1}\}_{k_2}$ will be denoted $\{p\}_{k_1,k_2}$. Brackets will be used to indicate the list structure of a message (*i.e.* $[p_1, p_2, \ldots]$).

The adversary in our system is a set of users and routers $A \subseteq N$. The adversary is active in the sense that the automata running on members of A are completely arbitrary. We call an agent a *compromised* if $a \in A$

3.2 Automata

We give the automata descriptions for the users and routers that are based on the Tor protocol [9]. We have simplified the protocol in several ways. In particular we don't perform key exchange, do not use a stream cipher, have each user construct exactly one circuit to one destination, do not include circuit teardowns, eliminate the final unencrypted message forward, and omit stream management and congestion control. We are also using circuit identifiers to mimic the effect of a timing attack. Section 4.7 discusses the effects of changing some of these features of our protocol.

During the protocol each user u iteratively constructs a circuit to his destination. u begins by sending the message $\{CREATE\}_{k_1}$ to the first router, r_1, on his circuit. The message is encrypted with a key, k_1, shared between u and r_1. r_1 identifies k_1 by repeatedly trying to decrypt the message with each one of its keys until the result is a valid control message. It responds with the message CREATED.

Given a partially-constructed circuit, u adds another router, r_i, to the end by sending the message $\{[\text{EXTEND}, r_i, \{\text{CREATE}\}_{k_i}]\}_{k_{i-1},\ldots,k_1}$ down the circuit. As the message gets forwarded down the circuit, each router decrypts it. r_{i-1} performs the CREATE steps described above, and then returns the message $\{\text{EXTENDED}\}_{k_{i-1}}$. As the message gets forwarded back up the circuit, each router encrypts it.

Link identifiers are used by adjacent routers on a circuit to differentiate messages on different circuits. They are only unique to the pair. Circuit identifiers are also included with each message and identify the circuit it is traveling on. They are unique among all circuits. Circuit identifiers are not used in the actual Tor protocol, and their only purpose here is to represent the ability of an adversary to insert and detect timing patterns in the traffic along a circuit. Since the model does not include time, but timing attacks are very real [16], this is a way to give the adversary this power. It has the advantages of making it clear when this power is used and of being easy to remove in future model adjustments.

The user automaton's state consists of a routing circuit, a number that identifies its circuit, and a number that indicates the state of that circuit. We consider the final router in the circuit to be the destination of the user. The user automaton runs two threads, one that is called upon receipt of a message and the other that is called at the start of execution. To be concise, we will express these in pseudocode rather than IO automata, but it should be noted that the state changes in a particular branch occur simultaneously in the automaton. $h(c, i)$ indicates the number of occurrences of the ith router in the first i routers of a circuit c. (u, r, i) denotes the ith key shared by user u and router r. The automaton for user u appears in Automaton 1.

The router automaton's state is a set of keys and a table, T, with a row for each position the router holds in a circuit. Each row stores the previous and next hops in the circuit, identifying numbers for the incoming and outgoing links, and the associated key. There is only one thread and it is called upon receipt of a message. We denote the smallest positive integer that is not being used on a link from r to q or from q to r as $minid(T, q)$. The automaton for router r appears in Automaton 2.

3.3 System execution

We use the standard notions of an *execution* and *fairness*. An execution corresponds to a possible run of the network given its initial state. Fairness in our model simply means that any message an automaton wants to send will eventually be sent and that every sent message is eventually received. Usually, the fairness condition makes it easier to design distributed algorithms; however, in our case, fairness makes it more difficult, because it restricts the executions that the adversary must consider when trying to break anonymity.

We also introduce the notion of a *cryptographic* execution. This is an execution in which no agent sends a control message encrypted with active keys it doesn't possess before it is sent that encrypted message. We will restrict our attention to such executions and must appeal to computational intractability to justify this restriction. Our encryption operation must only allow an attacker to output a control message in P encrypted with active keys it doesn't possess with negligible probability. This is reasonable because we can easily create a ciphertext space that is much larger than the rather limited control message space P. Note that this precludes the use of public key encryption to construct the onions because such messages can easily

Automaton 1 User u

1: $c \in \{(r_1, \ldots, r_l) \in R^l | \forall_i r_i \neq r_{i+1}\}$; init: arbitrary ▷ User's circuit
2: $i \in \mathbf{N}$; init: random ▷ Circuit identifier
3: $b \in \mathbf{N}$; init: 0 ▷ Next hop to build
4: **procedure** START
5: SEND$(c_1, [i, 0, \{\text{CREATE}\}_{(u,c_1,1)}])$
6: $b = 1$
7: **end procedure**
8: **procedure** MESSAGE(msg,j) ▷ $msg \in M$ received from $j \in N$
9: **if** $j = c_1$ **then**
10: **if** $b = 1$ **then**
11: **if** $msg = [i, 0, \text{CREATED}]$ **then**
12: $b++$
13: SEND$(c_1, [i, 0, \{[\text{EXTEND}, c_b, \{\text{CREATE}\}_{(u,c_b,h(c,b))}]\}_{(u,c_{b-1},h(c,b-1)),\ldots,(u,c_1,h(c,1))}])$
14: **end if**
15: **else if** $b < l$ **then**
16: **if** $msg = [i, 0, \{\text{EXTENDED}\}_{(u,c_{b-1},h(c,b-1)),\ldots,(u,c_1,h(c,1))}]$ **then**
17: $b++$
18: SEND$(c_1, [i, 0, \{[\text{EXTEND}, c_b, \{\text{CREATE}\}_{(u,c_b,h(c,b))}]\}_{(u,c_{b-1},h(c,b-1)),\ldots,(u,c_1,h(c,1))}])$
19: **end if**
20: **else if** $b = l$ **then**
21: **if** $msg = [i, 0, \{\text{EXTENDED}\}_{(u,c_{b-1},h(c,b-1)),\ldots,(u,c_1,h(c,1))}]$ **then**
22: $b++$
23: **end if**
24: **end if**
25: **end if**
26: **end procedure**

be constructed with the public keys of the routers.

Definition 1. An *execution* is a sequence of states of an IO automaton alternating with actions of the automaton. It begins with an initial state, and two consecutive states are related by the automaton transition function and the action between them. Every action must be *enabled*, meaning that the acting automaton must be in a state in which the action is possible at the point the action occurs. Because there are no internal actions in the automata, all actions are message sends or message receives. Frequently, we will treat an execution as a sequence of actions, because the states are implicit from these and the initial states.

Definition 2. A finite execution is *fair* if there are no actions enabled in the final state. Call an infinite execution *fair* if every output action that is enabled in infinitely many states occurs infinitely often.

Definition 3. An execution is *cryptographic* if no user or router sends a control message in P encrypted by a key it does not possess before receiving that message at least once. More formally, no router r sends a message $[n, \{p\}_{(u,q_1,n_1),\ldots,(u_k,q_k,n_k)}] \in M$, where $r \neq q_i$, for some i, and no user u sends a message $[n, \{p\}_{(v_1,r_1,n_1),\ldots,(v_k,r_k,n_k)}]$, $u \neq v_i$, for some i, before it receives a message of that form.

3.4 Distinguishability

The actions we want to be performed anonymously are closely related to the circuits the users try to construct during an execution.

Definition 4. A *configuration* $C : U \to \{(r_1, \ldots, r_l, n) \in R^l \times \mathbf{N}_+ | \forall_i r_i \neq r_{i+1}\}$ maps each user to the circuit and circuit identifier in his automaton state.

Automaton 2 Router r

$keys \in K^n$, where $n \geq |U| \cdot \lceil \frac{l}{2} \rceil$; init: arbitrary ▷ Private keys
$T \subset N \times \mathbf{N} \times R \times \mathbf{Z} \times \mathbf{Z}_{|keys|}$; init: $\emptyset$ ▷ Routing table
procedure MESSAGE($[i, n, p], q$) ▷ $[i, n, p] \in M$ received from $q \in N$
  if $[q, n, \emptyset, -1, k] \in T$ **then** ▷ In link created, out link absent
    if $\exists_{s \in R-r, b \in P} p = \{[\text{EXTEND}, s, b]\}_k$ **then**
      SEND($s, [minid(T, s), b]$)
      $T = T - [q, n, \emptyset, -1, k] + [q, n, s, -minid(T, s), k]$
    end if
  else if $[s, m, q, -n, k] \in T$ **then** ▷ In link created, out link initiated
    if $p = \text{CREATED}$ **then**
      $T = T - [s, m, q, -n, k] + [s, m, q, n, k]$
      SEND($s, [i, m, \{\text{EXTENDED}\}_k]$)
    end if
  else if $\exists_{m>0} [q, n, s, m, k] \in T$ **then** ▷ In and out links created
    SEND($s, [i, m, \{p\}_{-k}]$)
  else if $[s, m, q, n, k] \in T$) **then** ▷ In and out links created
    SEND($s, [i, m, \{p\}_k]$)
  else
    if $\exists_{k \in keys} p = \{\text{CREATE}\}_k$ **then** ▷ New link
      $T = T + [q, n, \emptyset, -1, k]$
      SEND($q, [i, n, \text{CREATED}]$)
    end if
  end if
end procedure

In our model, all messages are sent along links of a circuit; these messages are all circuit-creation messages and thus are entirely determined by the circuit, so the sender or receiver of a given message corresponds directly to the path of the circuit. Therefore, in order to prove that certain actions are performed anonymously in the network, we can just show that the adversary can never determine this circuit information. This is a possibilistic notion of of anonymity. We will do this by identifying classes of configurations among which an adversary cannot distinguish.

Because $i \in N$ only sees those messages sent to and from i, an execution of a configuration C may appear the same to i as a similar execution of another configuration D that only differs from C in parts of the circuits that are not adjacent to i and in circuit identifiers that i never sees. To be assured that i will never notice a difference, we would like this to be true for all executions of C that could occur. These are the fair, cryptographic executions of C, and likewise the execution of D should be fair and cryptographic.

We will say that these configurations are indistinguishable if, for any fair cryptographic execution of C, there exists a fair cryptographic execution of D that appears identical to i, *i.e.* in which i sends and receives what appear to be the same messages in the same order.

Agent i's power to distinguish among executions is weakened by encryption in two ways. First, we allow a permutation on (user,router,position) triples, which identify the keys in the system, to be applied to the keys of encrypted or decrypted messages in an execution. This permutation can map a key from any router other than i to any other key of any other router other than i, because i can only tell that it doesn't hold these keys. It can map any key of i to any other key of i, because i doesn't know for which users and circuit positions its keys will be used. Second, i cannot distinguish among messages encrypted with a key he does not possess, so we allow a permutation to be applied to control messages that are encrypted with a key that is not shared with i. This second requirement must be justified by the computational intractability of distinguishing between encrypted messages with more than a negligible probability in our cryptosystem.

Definition 5. Define D_A to be a relation over configurations that indicates which configurations are indis-

tinguishable to $A \subseteq N$. For configurations C, C', $C \sim_{D_A} C'$ if, for every fair cryptographic execution α of C, there exists some action sequence β such that the following conditions hold when C' is the initial state:

1. Every action of β is enabled, except possibly for actions performed by a member of A.
2. β is fair for all agents, except possibly those in A.
3. β is cryptographic for all agents.
4. Let Ξ be the subset of permutations on $U \times R \times \lceil \frac{l}{2} \rceil$ such that each element restricted to keys involving $a \in A$ is a permutation on those keys. We apply $\xi \in \Xi$ to the encryption of a message sequence by changing every list component $\{p\}_{(u,r,i)}$ in the sequence to $\{p\}_{\xi(u,r,i)}$.

   Let Π be the subset of permutations on $\bar{P}$ such that for all $\pi \in \Pi$:

   (a) π is a permutation on the set $\{\{p\}_{k_1,\ldots,k_i}\}_{p \in P}$

   (b) $\pi(\{p\}_{k_1,\ldots,k_i,k_a}) = \pi(\{p\}_{k_1,\ldots,k_i})$, where k_a is shared by the adversary

   We apply $\pi \in \Pi$ to a message sequence by changing every message $\{p\}_{k_1,\ldots,k_i}$ in the message sequence to $\pi(\{p\}_{k_1,\ldots,k_i})$.

   Then there must exist $\xi \in \Xi$ and $\pi \in \Pi$ that such that applying ξ and π to the subsequence of α corresponding to actions of A yields the subsequence of β corresponding to actions of A.

If $C \sim_{D_A} C'$, we say that C is *indistinguishable* from C' to A. It is clear that an indistinguishability relation is reflexive and transitive.

3.5 Anonymity and Unlinkability

The sender in this model corresponds to the user of a circuit, the receiver to the last router of the circuit, and the messages we wish to communicate anonymously are just the circuit control messages. The circuit identifiers allow the adversary to link together all the messages initiated by a user and attribute them to a single source. Therefore sender anonymity is provided to u if the adversary can't determine which circuit identifier u is using. Similarly, receiver anonymity is provided to r for messages from u if the adversary can't determine the destination of the circuit with u's identifier. Also, unlinkability is provided to u and r if the adversary can't determine u's destination.

Definition 6. User u has *sender anonymity* in configuration C with respect to adversary A if there exists some indistinguishable configuration C' in which u uses a different circuit identifier.

Definition 7. Router r has *receiver anonymity* on user u's circuit, in configuration C, and with respect to adversary A, if there exists some indistinguishable configuration C' in which a user with u's circuit identifier, if one exists, has a destination other than r.

Definition 8. User u and router r are *unlinkable* in configuration C if there exists some indistinguishable configuration C' in which the destination of u is not r.

4 Indistinguishable Configurations

Now we will show that sometimes the adversary cannot determine the path or identifier of a circuit. More specifically, an adversary can only determine which user or router occupies a given position in a circuit when the adversary controls it or a router adjacent to it on that circuit. Also, when the adversary controls no part of a circuit it cannot determine its identifier. In what follows, let C be some configuration.

4.1 Message Sequences

To start, we observe that, in spite of the arbitrary actions of the adversary, the actions of the uncompromised users and routers are very structured. The protocol followed by the user and router automata defines a simple sequence of message sends and receives for every circuit. A user or router will only send messages from the part of such a sequence consisting of its actions.

The user automaton gives this subsequence for users. It consists of messages between the user and the first router on its circuit, and is parameterized by the user and the system configuration. We will refer to this sequence as $\sigma_U(u, C)$. We denote the number of occurrences of the ith router in the first i routers of u's circuit c by $h(c, i)$. For convenience, we define the function $k(u, C, i) = (u, C_i(u), h(C(u), i))$, which is the key shared between u and its ith router in C. Since the user automaton ignores the circuit identifier on received messages, we use an asterisk to indicate that any value is valid. Let $\sigma_U(u, C)$ be:

Step	From	To	Message
1	u	$C_1(u)$	$[C_{l+1}(u), 0, \{\text{CREATE}\}_{k(u,C,1)}]$
2	$C_1(u)$	u	$[*, 0, \text{CREATED}]$
3	u	$C_1(u)$	$[C_{l+1}(u), 0, \{[\text{EXTEND}, C_2(u), \{\text{CREATE}\}_{k(u,C,2)}]\}_{k(u,C,1)}]$
4	$C_1(u)$	u	$[*, 0, \{\text{EXTENDED}\}_{k(u,C,1)}]$
$1 + 2i$	u	$C_1(u)$	$[C_{l+1}(u), 0, \{[\text{EXTEND}, C_{i+1}(u), \{\text{CREATE}\}_{k(u,C,i+1)}]\}_{k(u,C,i),\ldots,k(u,C,1)}]$
$2 + 2i$	$C_1(u)$	u	$[*, 0, \{\text{EXTENDED}\}_{k(u,C,i),\ldots,k(u,C,1)}]$

$2 \leq i < l$

Lemma 1. *A user u is enabled to send a message in an action sequence under configuration C iff the following conditions are satisfied:*

1. *The send appears in $\sigma_U(u, C)$.*
2. *The prefix of $\sigma_U(u, C)$ ending before the send has appeared in the sequence.*
3. *This prefix is the longest such prefix to appear.*

□

Similarly, the router automaton defines the action sequence that a router performs during the creation of a circuit. A different sequence exists for every router r, user u, circuit position $1 \leq i \leq l$, system configuration C, and link identifiers $m, n, p \in \mathbf{N}$. We will denote a particular sequence $\sigma_R(r, C, u, i, m, n)$. Frequently we will drop parameters that we don't care about, for example, referring to $\sigma_R(r, C, u, i)$ when the specific link identifiers don't matter, and may abuse this notation by treating it as one sequence rather than a family of sequences. We use $k(u, C, i)$ as before. The sequence $\sigma_R(r, C, u, i, m, n)$ is:

Step	From	To	Message
1	$C_{i-1}(u)$	r	$[j_1, n, \{\text{CREATE}\}_{k(u,C,i)}]$
2	r	$C_{i-1}(u)$	$[j_1, n, \text{CREATED}]$
3	$C_{i-1}(u)$	r	$[j_2, n, \{[\text{EXTEND}, C_{i+1}(u), \{\text{CREATE}\}_{k(u,C,i+1)}]\}_{k(u,C,i)}]$
4	r	$C_{i+1}(u)$	$[j_2, m, \{\text{CREATE}\}_{k(u,C,i+1)}]$
5	$C_{i+1}(u)$	r	$[j_3, \text{CREATED}]$
6	r	$C_{i-1}(u)$	$[j_3, \{\text{EXTENDED}\}_{k(u,r,i)}]$

Using the σ_R sequences, we can characterize which messages a router can send at any point in an action sequence. Let α be a finite execution, and τ_i be the length i prefix of some sequence $\sigma \in \sigma_R(r)$. We say that τ_i has *occurred* in α if, by the end of the sequence, r has performed the first i steps in σ. This happens when τ_i is the longest prefix of σ that appears as a subsequence of α, and, at the point at which step 1 (4) occurs in α, n (m) must be the smallest number not yet in r's table as an entry to or from c_{i-1} (c_{i+1}), the router that sent (received) the message in step 1 (4).

Lemma 2. *For r to be enabled to send a message μ at the end of α, one of three cases must apply for some $\sigma \in \sigma_R(r)$:*

1. *The message is part of the circuit building protocol. Sending μ is the $(2i)^{th}$ step in σ, $1 \leq i \leq 3$. Then τ_{2i-1} must occur in α and τ_{2i} must occur in the execution created when μ is appended to α.*

2. *The message is a forward up the circuit. $\mu = [m, p]$. r is to be sending μ to c_{i+1}. σ has occurred in α. The link identifiers to c_{i-1} and c_{i+1} are n and m, respectively. α contains the action of the message $[n, p]$ being sent to r from c_{i-1} after σ occurs. Also, the number of such receives from c_{i-1} is greater than the number of sends of μ to c_{i+1} that happen after σ occurs.*

3. *The message is a forward down the circuit. $\mu = [n, p]$. r is to be sending μ to c_{i-1}. σ has occurred in α. The link identifiers to c_{i-1} and c_{i+1} are n and m, respectively. α contains the action of the message $[m, p]$ being sent to r from c_{i+1} after σ occurs. Also, the number of such receives from c_{i+1} is greater than the number of sends of μ to c_{i-1} that happen after σ occurs.*

□

We can use these lemmas to partition the actions performed by an agent in an execution of configuration C. We will use these partitions to construct executions of indistinguishable configurations and prove that they satisfy the requirements of Definition 5.

For a user u we create two partitions. The first is formed by the maximal prefix of $\sigma_U(u, C)$ such that each receive in the partition causes the b variable of u's state to be incremented. The condition on the receives is required for a unique maximal prefix to deal with the case that an adversary sends sequence responses multiple times. The second partition is formed from all of u's other actions. By Lemma 1 this is composed of receiving unnecessary messages due to adversarial actions, and we will call this the "junk" partition.

For a router r, we create a partition for each entry in its routing table at any point in the execution and an extra junk partition. For a given routing table entry we create a partition out of the maximum-length subsequence of some $\sigma_R(r)$ sequence, say, σ, such that each receive modifies the same entry in the routing table. We also include every send and receive of a forward performed using that entry. This partition is said to be associated with σ. Every other action done by the router is put in a junk partition, and, by Lemma 2, this partition is composed only of receives.

4.2 Indistinguishable Users

Now we prove that an active adversary cannot determine which user creates a given circuit unless the first router on that circuit is controlled by the adversary or the owners of all the other circuits have been determined. That is, an adversary cannot distinguish between a configuration C and the configuration C' that is identical to C except for two circuits with uncompromised first routers that are switched between their owners. In order to do so, we must show that, for any fair, cryptographic execution of C, there exists some action sequence of C' satisfying the indistinguishability requirements of Definition 5. To do so, we simply swap between the switched users the messages that pass between them and the first routers on their circuits and switch the encryption keys of these messages.

Theorem 1. *Say there are two distinct users, u, v, such that neither they nor the first routers in their circuits are compromised (that is, in A). Let C' be identical to C except the circuits of users u and v are switched. C is indistinguishable from C' to A.*

Proof. Let α be a fair, cryptographic execution of C. To create a possible execution of C', first construct α' by replacing any message sent or received between u (v) and $C_1(u)$ ($C_1(v)$) in α with a message sent or received between v (u) and $C_1(u)$ ($C_1(v)$). Then let ξ be the permutation that sends u to v and v to u and other users to themselves. Create β by applying ξ to the encryption keys of α'.

1. *Every action by an agent in $N \setminus A$ in β is enabled.* It is easy to see that all receives are enabled in β since sends and corresponding receives are modified together.

   For any user $w \notin \{u, v\}$, all messages in $\sigma_U(w, C)$ go to or from w, so none are added or removed from α in α'. Also none of the messages in this sequence would be modified by ξ because they are encrypted

with a key of w, and ξ doesn't convert messages in α' to be messages of $\sigma_U(w, C)$. Therefore if a message is enabled to be sent from w in β it was enabled in α.

For user u, when u sends a message to $C_1(v)$ in β, it corresponds to v sending a message to $C_1(v)$ in α. v is enabled to do so in α so at that point it has sent and received exactly those messages of $\sigma_U(v, C)$ necessary to enable that send. In β we have changed those messages to be messages between u and $C_1(v)$ while modifying the encryption keys, so the necessary $\sigma_U(u, C')$ messages have appeared to enable the send. No additional messages in $\sigma_U(u, C')$ could have appeared in β since u and v do not communicate on link identifier 0 in α. Therefore u is enabled to send the message. A similar argument works for v.

For a router $r \notin A \cup \{C_1(u), C_1(v)\}$, the only change in messages to or from r between α and β is from the permutation ξ applied to the encryption keys of the messages. Applying ξ preserves the occurrence of some prefix of $\sigma_R(r, C', w)$ at any point in the execution, for any $w \notin \{u, v\}$. For $w = u$, applying ξ turns the occurrence of some $\sigma_R(r, C, u)$ into an occurrence of $\sigma_R(r, C', v)$, and vice versa for $w = v$. It also preserves the appearance of messages to forward and the actual forwarding. Thus any action performed by r in β is enabled because it corresponds to a similar enabled action in α.

Now consider a message μ sent from $C_1(u)$ in β.

It may be that μ is part of a $\sigma_R(C_1(u), C, w)$ sequence for some $w \notin \{u, v\}$ in α. Then μ is enabled in β since none of the messages in $\sigma_R(C_1(u), C, w)$ come from u or v and none involve u or v in the encryption keys, so all exist in β that did in α and no additional ones do. It could also be that, in α, μ is a forward in some circuit not belonging to u or v. Then μ is still enabled in β for a similar reason, recognizing that although it might involve the encryption keys of u or v, the content of messages is ignored in forwards.

Another case is that, in α, μ is part of some $\sigma_R(C_1(u), C, u, i)$. Then in β, μ is part of $\sigma_R(C_1(u), C', v, i)$. This is because every message from u to $C_1(u)$ is changed to a message from v to $C_1(u)$ and every encryption key involving u changes to one involving v. It is clear that consistently replacing the user in the encryption keys in a σ_R sequence and (when $i = 1$) the previous hop from u to v, as is done to create β, transforms a $\sigma_R(C_1(u), C, u, i)$ sequence into a $\sigma_R(C_1(u), C', v, i)$ sequence. No additional messages can enter into this sequence in β because they must be encrypted with a key of v, and any such message will have appeared with a key of u in α and will have filled the same spot in the $\sigma_R(C_1(u), C, u, i)$ sequence there. Thus μ is enabled in β. Also, for similar reasons, if μ is a forward in u's circuit in α, then it is a forward for v's circuit in β.

The final case is when, in α, μ is in a $\sigma_R(C_1(u), C, v)$ sequence or is a forward on v's circuit. Since v does not communicate directly with $C_1(u)$ as a user in α, it must be that $C_1(u)$ is some intermediate router. Then the only changes to the $\sigma_R(C_1(u), C, v)$ messages in α are the encryption keys, which are applied consistently to all the sequence messages and are not interfered with by messages in α already using the target keys since they are also modified. Therefore if μ corresponds to a $\sigma_R(C_1(u), C, v)$ send in α, it is a $\sigma_R(C_1(u), C', u)$ message enabled in β. Also, for similar reasons, if μ was a foward in v's circuit in α, it is an enabled forward on u's circuit in β.

A similar argument works for $C_1(v)$.

2. *β is fair for agents in $N \setminus A$.*

   For any user $w \notin \{u, v\}$, every $\sigma_U(w, C')$ message received in β in its non-junk partition is the same message in α. Therefore every send w is enabled to perform in β it is enabled to perform in α. Since α is fair for w so is β.

   Now consider u. The transformation properly changes the messages from $C_1(v)$ to v in $\sigma_U(u, C)$ to messages sent to u that are in the same position in the $\sigma_U(u, C')$ sequence. No extra messages can appear since they must be encrypted using u's keys, and then they would have been encoded with v's keys in α and been part of the $\sigma_U(v, C)$ sequence there. Therefore every send that u is enabled to

perform in β, v is enabled to perform in α. Since α is fair for v, then β is fair for u. A similar argument works for v.

For router $r \notin A \cup \{C_1(u), C_1(v)\}$ to be enabled to perform a send in β but not α, there must be a message in some sequence $\sigma_R(r, C', w, i)$ that r receives in β but doesn't in the corresponding sequence in α. This cannot be for any $w \notin \{u, v\}$, since the messages in this σ_R are not modified, except possibly the content of forwards which doesn't affect their validity. All messages in β that are to r and are in some $\sigma_R(r, C', u)$ are also are sent to r in α and are part of some $\sigma_R(r, C', v)$. Therefore if such a message enables r to send something in β there exists a similar message enabling r to send something in α. Also forwards along u's circuit in β exist as forwards along v's circuit in α. A similar argument works for messages of some sequence $\sigma_R(r, C', v)$.

For $C_1(u)$ to be enabled to perform a send in β but not α, there must be a message in some sequence $\sigma_R(C_1(u), C', w)$ or forward that $C_1(u)$ receives in β but doesn't in α. There can not be such a message in the $\sigma_R(C_1(u), C', w)$ sequence for any $w \notin \{u, v\}$, since the messages in this sequence are not modified in the transformation and no new messages encrypted with w's key are created. Also the sender and recipient of forwards aren't modified, and the content of forwards which doesn't affect their validity. Now suppose $w = v$. For any message that appears at the end of some $\sigma_R(C_1(u), C', v, i)$ in β that $C_1(u)$ doesn't respond to there must not be an analogous message in $\sigma_R(C_1(u), C, u, i)$ in α or $C_1(u)$ would be enabled at that point as well. But this message must be encrypted with v's keys and would be modified by the ξ_U permutation and thus play the same role for $C_1(u)$ in α. Again, the content of forwards doesn't matter and any forward on v's circuit in β corresponds to a forward on u's circuit in α. A similar argument works for the case $w = u$. Therefore every send enabled for $C_1(u)$ in β is enabled in α, and β is fair for $C_1(u)$. A similar argument works for $C_1(v)$.

3. *β is cryptographic.*

   We've already shown that uncompromised routers and users perform enabled actions in β. Since the automatons only allow sending messages encrypted with keys the agent doesn't possess after receiving them, the actions of these agents do not prevent β from being cryptographic. For a compromised user or router, let's say a message encrypted with a foreign key is sent before being received at least once. If the encryption key doesn't involve u or v, then the same message gets sent in α before being received, contradicting the fact that α is cryptographic. If the key does involve u, then in α it involves v, in which case if the message is received in α beforehand, it must have received it in β since the key permutation takes v to u. Likewise for messages encrypted with one of v's keys. The fact that α is cryptographic then implies that β is cryptographic.

4. *We can find a $\xi \in \Xi$ and $\pi \in \Pi$ that turn α into a sequence that agrees with β in all the adversary actions.*

   ξ is simply the user permutation used to create β, transposing users u and v, and π is the identity on all messages. Applying these to α yields a sequence that agrees with β everywhere except for messages between u (v) and $C_1(u)$ ($C_1(v)$), which we assumed are not adversarial.

□

4.3 Indistinguishable Routers

Now we prove that an adversary cannot determine an uncompromised router on a given circuit unless it controls the previous or next router on that circuit. More formally, assume that the $(i-1)$st, ith, and $(i+1)$st routers of a user u's circuit in some configuration C are not compromised. We will show that C is indistinguishable from configuration C', where C' is identical to C except the ith router of u's circuit has been arbitrarily changed. The proof is similar to that of Theorem 1, although it is complicated by the fact that the identities of routers in a circuit are included in multiple ways in the circuit creation protocol. Given an execution of C, we identify those message that are part of the circuit creation sequence of the modified

circuit and then change them to add a different router in the ith position. Then we show that, in the sense of Definition 5, from the adversary's perspective this sequence is indistinguishable from the original and could be an execution of C'.

Theorem 2. *Say there is some user $u \notin A$ such that u's circuit in C contains three consecutive routers, $r_{i-1}, r_i, r_{i+1} \notin A$. Let C' be equal to C, except r_i is replaced with r'_i in u's circuit, where $r'_i \notin A \cup \{r_{i-1}, r_{i+1}\}$. C' is indistinguishable from C to A. The same holds for uncompromised routers (r_i, r_{i+1}) if they begin u's circuit and are replaced with (r'_i, r_{i+1}), or (r_{i-1}, r_i) if they end u's circuit and are replaced with (r_{i-1}, r'_i).*

Proof. Let α be some fair cryptographic execution of C, and let $h(C(u), i)$ denote the number of occurrences of the ith router in the circuit $C(u)$ among the first i routers. We modify α in steps to create an indistinguishable sequence β:

1. Replace all message components of the form $[\text{EXTEND}, r_i, \{\text{CREATE}\}_{u,r_i,h(C(u),i)}]$ with $[\text{EXTEND}, r'_i, \{\text{CREATE}\}_{u,r'_i,h(C'(u),i)}]$.

2. Consider the partition of router r_{i-1}'s actions that are associated with a $\sigma_R(r_{i-1}, C, u, i-1)$ sequence. Replace all messages in this partition that are to and from r_i with the same messages to and from r'_i. Modify the link identifiers on these messages so that they are the smallest identifiers in use between r_{i-1} and r'_i at that point in α. Increase link identifiers that are in use between r_{i-1} and r'_i to make room for these new connections and decrease link identifiers that are in use between r_{i-1} and r_i to fill in the holes created by the removed connections. Perform similar modifications for routers r_i and r_{i+1}.

3. Replace all encryption keys of the form $(u, r_i, h(C(u), i))$ with $(u, r'_i, h(C'(u), i))$. Increment as necessary the third component of the encryption keys used between u and r'_i to take into account that r'_i appears once more in $C'(u)$ than it does in $C(u)$. Also decrement as necessary the third component of the keys used between u and r_i to take into account that r_i appears once less in $C'(u)$ than it does in $C(u)$.

Now we show that the action sequence thus created, β, is a fair cryptographic execution of C':

1. *Every action by an agent in $N \setminus A$ in β is enabled.*

   It is easy to see that all receives are enabled in β since sends and corresponding receives are modified together.

   Our strategy to show that all sends in β are enabled will be to consider the separate non-junk partitions of α after the transformation. First we will show that no sends from uncompromised agents appear in β outside of these transformed partitions. Then we show that any given non-junk partition of α is transformed into a subsequence that is "locally" enabled under C'. A user (router) action sequence is locally enabled if each send satisfies the conditions of Lemma 1 (2) applied just to that sequence. Then we show that it is "globally" enabled in the sense that the sends in the transformed user (router) partition continue to satisfy Lemma 1 or Lemma (2), respectively, when considered over the entire sequence β.

   It is easier to proceed this way since going from a locally to globally enabled sequence just requires that certain actions *don't* exist in the larger sequence. For users, none of the sends in the transformed non-junk partition can appear again in the larger sequence between being enabled and being sent in the partition. This must also be true for transformed router partitions, and additionally the link identifiers used must be unique and minimal at the point of link creation. It should be easy to see that a locally enabled action sequence satisfying these global conditions contains only enabled sends in β, via Lemmas 1 and 2.

   The fact that there are no sends from uncompromised agents in β outside of the transformed non-junk α partitions helps us prove that actions are globally enabled. By inspecting the three changes made to α, it is clear that no actions are added or deleted from α, and that sends (receives) in α are sends

(receives) in β. Since every send in α from an agent $a \in N \setminus A$ is part of one of its non-junk partitions, every send by an uncompromised agent in β is part of one of the transformed partitions.

We can use this to show that a given sequence is globally enabled. If the sequence is a locally enabled transformed user partition, it is automatically globally enabled because there are no sends outside the partition to interfere with it. Similarly, for locally enabled transformed router partitions, we automatically satisfy the send non-inteference property in β as long as we satisfy the second requirement on the link identifiers.

This second requirement for routers is slightly simpler to achieve by noting that all CREATE messages in β were transformed from CREATE messages in α. Therefore we only need to show that the link IDs used in β are unique and minimal among the link creations in α after transformation.

For user $v \neq u$ the non-junk partition has not been modified therefore by Lemma 1 every send is locally enabled in β. Therefore every action by v is enabled.

Now consider the user u's non-junk partition in α. We've modified steps $2i-1$,$2i$,$2i+1$, and $2i+2$ as necessary to change the $\sigma_U(u, C)$ prefix to a $\sigma_U(u, C')$ prefix. All these are enabled by Lemma 1 so this is locally enabled. No sends appear outside of this transformed partition in β. Thus the partition is globally enabled.

Now consider a router $r \notin \{r_{i-1}, r_i, r'_i, r_{i+1}\}$ and a partition of r in α. The partition consists of a prefix of some $\sigma_R(r, C)$ sequence and possibly some forwards. The only changes made to the partition are key relabelings and some modification of the messages of forwards. The relabeling turns the $\sigma_R(r, C)$ prefix into some $\sigma_R(r, C')$ prefix of the same length, so sends in this sequence are locally enabled. Forwards are enabled regardless of content, so they are also locally enabled. No link identifiers of r have changed, so they are still unique and minimal, so the whole partition is globally enabled.

Now consider r_{i-1}. Take some non-junk partition of α that is not associated with a sequence to u as the $(i-1)$th circuit router, that is, that is associated with a $\sigma_R(r_{i-1}, C, w, j)$ sequence, $w \neq u \vee j \neq i-1$. For the same reasons as the preceding case, it is transformed into a sequence associated with a $\sigma_R(r_{i-1}, C', w)$ sequence. Thus it is locally enabled. The partition that is a prefix of $\sigma_R(r_{i-1}, C, u, i-1)$ can be seen by inspection to be modified to be a locally enabled sequence associated with $\sigma_R(r_{i-1}, C', u, i-1)$. The link identifiers used in every transformed partition of r_{i-1} are unique and minimal in β because the original partitions had unique and minimal IDs in α, we haven't changed the IDs or neighbors of any partitions not connecting with r'_i or r_i, we have changed the ID in partitions connecting with r'_i or r_i to make IDs unique and minimal after changing a partition to connect with r'_i instead of r_i. Thus the whole sequence is globally enabled. Similar arguments work for r_{i+1}, r_i and r'_i.

2. *β is fair for agents in $N \setminus A$.*

   To show this we will again consider the transformed partitions of α. We have shown that they form enabled sequences, and now need to show that no messages from the transformed junk partition belong in these sequences. For users, this means that the next step in a transformed σ_U partition isn't received. For routers, it means that the next step in a transformed σ_R partition isn't received, no new forwards on a created circuit are received, and that no new valid CREATE messages are received.

   Consider a user $w \neq u$. Every receive action by w in β is from a receive action by w in α. The messages of w's receives are never modified to use one of w's keys, so a message encrypted with w's keys in β uses the same keys in α. Also the content of an encrypted message is never changed to be a message that appears in $\sigma_U(w, C')$. Therefore any receive that is a step of $\sigma_U(w, C')$ in β is the same step in $\sigma_U(w, C)$ in α. Therefore β is fair for w.

   Consider user u. As shown, the transformed non-junk partition in α is a locally enabled sequence in β. For u to have an unperformed enabled action in β, the next message in the $\sigma_U(u, C')$ sequence must come from the junk sequence and be unanswered. All the receives that are the same between $\sigma_U(u, C)$ and $\sigma_U(u, C')$ are left unchanged in β, so one of these cannot be the unanswered step. The received

mesages that are different between $\sigma_U(u, C)$ and $\sigma_U(u, C')$ are in steps $(2 + 2j)$, $i \leq j \leq l$. These only differ in the encryption keys in such a way that the transformation applied to α takes every $(2 + 2j)$th step in $\sigma_U(u, C)$ to the $(2 + 2j)$th step in $\sigma_U(u, C')$. Thus an enabling receive in β is such a receive in α. Therefore β is fair for u.

Now consider a router $r \notin \{r_{i-1}, r_i, r'_i, r_{i+1}\}$. For a given transformed partition, no new messages of the associated $\sigma_R(r)$ sequence can appear in β since $\sigma_R(r)$ messages are all encrypted for r and we have created no such messages nor modified their content in such a way as to create a new message in the $\sigma_R(r)$ sequence. For forwards, first we recognize that the transformation maintains source and link ID consistency for r in the sense that if we were to group r's receives in α by their source and link ID the transformed groups would be the same as the same groups created in β. Therefore for an incoming message to be transformed into a forward on a created circuit, it must previously be sent with the link identifiers of the incoming link, but since content in forwards doesn't matter, this would be a forward in α as well. Finally there are no valid CREATE messages received in β that aren't received in α. No new messages are sent to r, no messages are transformed into a CREATE, no keys have been modified to belong to r, and r's link IDs have been consistently changed. Therefore β is fair for r.

Now consider r_{i-1}. For a transformed partition of r_{i-1} say that some receive extends the associated $\sigma_R(r_{i-1})$ or acts as a forward on the created circuit. This receive must be from the junk partition of r_{i-1} since we have shown its that non-junk partitions form enabled sequences. This message can't be from a router not in $\{r_i, r'_i\}$ because all such messages existed in α with the same link ID, source, and destination, and the content is either the same or is a forward, which would still be a forward in α. It can't come from r_i. This router is uncompromised and therefore properly uses link identifiers. If the message were to be part of the associated $\sigma_R(r_{i-1})$ sequence, it would exist in α with an ID identical to that in use by r_{i-1} and r_i in the sequence and with the same content, so this can't be the case. If the message were to be a forward, again it would exist in α with a link ID in use between r_{i-1} and r_i in the partition, and would therefore be a forward in α as well. Similar arguments work for messages from r'_i. Finally, no new partitions can be created, since CREATE messages to r_{i-1} on unique link IDs in β are the same in α. Therefore, β is fair for r_{i-1}. Similar arguments work for r_{i+1}.

Now consider r_i. Because only link identifiers to r_{i-1} and r_{i+1} have been changed, and those routers are uncompromised, all messages in β to r_i from a given router and with a given link ID are transformed from all message in α from the same router and of a given (possibly different) link ID. Thus for a message receive to act as the next step in the associated $\sigma_R(r_i)$ or to act as a new forward, it must have been sent in α on the link ID in α of that partition. Since messages aren't redirected to r_i and senders aren't changed it must have been sent in α by the same sender. Since content doesn't change in the $\sigma_R(r_i)$ messages and doesn't matter in forwards this message would perform the same function in α, contradicting the fairness of α. No new partitions can be created because new CREATE messages aren't made by the transformation, senders are the same, and link identifiers are renumbered in such a way that distinct link IDs from a router in α are distinct in β. Therefore β is fair for r_i.

A similar argument works for r'_i over its partitions in α, but we do reassign a partition of r_i to r'_i, which we must also consider. Notice that the messages redirected to r'_i exist on unique link IDs in β with r_{i-1} and r_{i+1} in β. Therefore these cannot enable actions on the other transformed partitions, and vice versa. Also no junk messages of r'_i can enable actions in this transformed partition because the connecting routers, r_{i-1} and r_{i+1}, are uncompromised and will have sent these messages on a link ID that is different from the ID of the transformed new partition in β. Finally, we show that the only valid CREATE messages received by r'_i in β are those in transformed partitions of α. Every CREATE in β is a CREATE in α. Every valid CREATE to r'_i in α becomes a valid CREATE in β because it is part of a transformed partition and we have shown that these become enabled sequences. The only messages redirected to r'_i belong to r_i's reassigned partition, which forms a fair sequence in α and maintains this after transformation. The final possibility for a new CREATE is a CREATE message from r'_i's junk partition that was sent to r'_i in α but was encrypted with a key of r_i, which then gets changed in the transformation. The only such message is $\{\text{CREATE}\}_{u,r_i,h(C(u),i)}$.

Only r_{i-1} could send this message in α. It would only do this if it were to receive the message $\{\text{EXTEND}, r'_i, \{\text{CREATE}\}_{u,r_i,h(C(u),i)}\}_{u,r_{i-1},h(C(u),i-1)}$, which u never sends in α. Again by the cryptographic property of α r_{i-1} never sends this CREATE to r'_i in α. Thus no valid CREATE messages are received by r'_i in β that are not transformed from partitions in α, which we have shown are fair. Therefore β is fair for r'_i.

3. *β is cryptographic.*

   For uncompromised routers the fact that all sends are enabled in β guarantees cryptographic sends since the protocol ensures this property. Compromised routers send and receive all the same messages, but to which the transformation function has been applied. Therefore since α is cryptographic, β is.

4. *We can find key and message permutations that turn β into a sequence that agrees with α in all adversary actions.*

   No messages are redirected towards or away from $a \in A$ when constructing β. We apply the message permutation to β of transposing $\{[\text{EXTEND}, r'_i, \{\text{CREATE}\}_{u,r'_i,h(C'(u),i)}]\}_{k_1,\ldots,k_j}$ and $\{[\text{EXTEND}, r_i, \{\text{CREATE}\}_{u,r_i,h(C(u),i)}]\}_{k_1,\ldots,k_j}$, $1 \leq j \leq l$, where k_j isn't shared by the adversary. We also apply the key permutation that sends $(u, r'_i, h(C'(u), i))$ to $(u, r_i, h(C(u), i))$ and undoes the renumbering of the r'_i and r_i keys. Then the subsequence of actions by s in β is identical to the subsequence in α.

□

4.4 Indistinguishable Identifiers

Theorem 3. *Say there is some uncompromised user u such that all routers in $C(u)$ are uncompromised. Then let C' be a configuration that is identical to C, except that u uses a different circuit identifier. C' is indistinguishable from C to A.*

Proof. Let α be a fair, cryptographic execution of C. To create β, simply change every occurrence of u's circuit identifier in C ($C_{l+1}(u)$) to its identifier in C'. β is enabled, fair, and cryptographic for C' because no message containing $C_{l+1}(u)$ gets sent to the adversary in α and the protocol itself ignores circuit identifiers except to forward them on. It appears the same to A for the same reason. □

4.5 Distinguishable Configurations

It is easy to see that the relation D_A, when restricted to the transitive closure of pairs that are indistinguishable by Theorems 1, 2, and 3, is symmetric and therefore forms an equivalence relation. We introduce some notation to conveniently refer to such configurations.

Definition 9. For configurations C and D, we say that $C \approx_{D_A} D$ if C and D are related by a chain of configurations that are indistinguishable by Theorems 1, 2, and 3.

We can easily tell which configurations are in the same equivalence class using the following function. It reduces a circuit to an identifier, the compromised positions, and the positions adjacent to compromised positions.

Definition 10. Let $\rho : U \times N^l \times \mathbf{N}_+ \times \mathcal{P}(N) \to \mathbf{N} \times \mathcal{P}(N \times \mathbf{N}_+)$ be:

$$\rho(u, c, A) = \begin{cases} (c_{l+1}, \{(r, i) \in N \times \mathbf{N}_+ | c_i = r \wedge (c_{i-1} \in A \vee c_i \in A \vee c_{i+1} \in A)\}) & \text{if } c_i \in A \text{ for some } i \\ (0, \emptyset) & \text{otherwise} \end{cases}$$

In the preceding let c_0 refer to u.

We overload this notation and use $\rho(C)$ to refer to the multiset formed from the circuits of configuration C adjoined with their user and reduced by ρ. That is, $\rho(C) = \{\rho(u, C(u), A) | u \in U\}$. It is not hard to see that ρ captures the indistinguishable features of a configuration according to Theorems 1, 2, and 3.

Proposition 1. *Let C and D be configurations. $C \approx_{D_A} D$ if and only if $\rho(C) = \rho(D)$.*

Now we show that the equivalence relation is in fact the entire indistinguishability relation and that Theorems 1, 2, and 3 characterize which configurations are indistinguishable. The reason for this is that an adversary can easily determine which entries in the compromised routers belong to the same circuits and what positions they hold in those circuits. The adversary links together entries in its routers by using the circuit identifiers that are uniquely associated with each circuit. And since circuits have a fixed length compromised routers can determine their position in the circuit by counting the number of messages received after the circuit entry is made.

Theorem 4. *Configurations C and D are indistinguishable only if $C \approx_{D_A} D$.*

Proof. Suppose that C and D are not in the same equivalence class. Let the adversary run the automata prescribed by the protocol on the agents it controls. Let α be a fair, cryptographic execution of C and β be a fair, cryptographic execution of D.

Partition the adversary actions of α into subsequences that share the same circuit identifier. There is at most one such partition for each circuit. Circuit positions that are created in the same partition belong to the same circuit. In each partition the adversary can determine the absolute location of a circuit position filled by a given compromised agent a by counting the total number of messages it sees after the initial CREATE. Clearly A can also determine the agents that precede and succeed a on the circuit and the circuit identifier itself. Therefore A can determine the reduced circuit structure $\rho(C)$ from α.

The adversary can use β in the same way to determine $\rho(D)$. By Proposition 1, $\rho(C) \neq \rho(D)$, so A can always distinguish between C and D. □

4.6 Anonymity

The configurations that provide sender anonymity, receiver anonymity, and unlinkability follow easily from Theorems 1, 2, 3, and 4.

Corollary 1. *User u has sender anonymity in configuration C with respect to adversary A if and only if at least one of the following cases is true:*

1. *u and $C_1(u)$ are uncompromised, and there exists another user $v \neq u$ such that v and $C_1(v)$ are uncompromised.*
2. *u and $C_i(u)$ are uncompromised, for all i.*

□

Corollary 2. *Router r has receiver anonymity on u's circuit, in configuration C, and with respect to adversary A if and only if at least one of the following cases is true:*

1. *u, r, and $C_{l-1}(u)$ are uncompromised, and there exists another router $q \neq r$ such that q is uncompromised.*
2. *u and $C_i(u)$ are uncompromised, for all i.*

□

Corollary 3. *User u and router r are unlinkable in configuration C with respect to adversary A if and only if at least one of the following cases apply:*

1. *u, r, and $C_{l-1}(u)$ are uncompromised, and there exists another router $q \neq r$ such that q is uncompromised.*
2. *u and $C_1(u)$ are uncompromised. There exists another user $v \neq u$ such that v and $C_1(v)$ are uncompromised. $C_l(v) \neq r$, or $C_{l-1}(v)$ and r are uncompromised and there exists another router $q \neq r$ such that q is uncompromised.*

□

4.7 Model Changes

We chose the described protocol to balance two goals. The first was to accurately model the Tor protocol . The second was to make it simple so that it could be analyzed, and also so that the main ideas of the analysis weren't unnecessarily complicated. Our results are robust to changes of the protocol, however. We can make the protocol simpler by removing multiple encryption and the circuit identifiers without weakening the indistinguishability results. In the other direction, we can make it more complicated with a stream cipher and multiple circuits per user without weakening the distinguishability results.

Multiple encryption is not necessary for the distinguishability theorems, and therefore the anonymity and unlinkability results. Consider a single-encryption protocol in which the user only encrypts each message with the key of the last router added to the circuit. Messages aren't encrypted or decrypted as they pass up and down a circuit. The adversary still is not able to determine parts of a circuit that aren't adjacent to a compromised agent. The proof of this under multiple encryption did not use the changing representation of messages going along a circuit, and only relied on the last key of the multiple encryption to hide the content of messages. Single encryption does allow the adversary to easily link entries in his routers by sending messages along the circuit. This power is already available in our model from circuit identifiers, though.

The circuit identifiers themselves are not actually necessary either. For any entry in a compromised router a, the adversary can simply wait until the circuit is created, and then send k_a dummy messages up the circuit, where k_a is some number unique to a. The first compromised router up the circuit will receive k_a messages that necessarily came from a because the circuit-building protocol will have finished. Linking entries this way is equivalent to using circuit identifiers because after it is done the adversary can easily simulate the presence of circuit identifiers on all the messages it receives.

Stream ciphers are used in the Tor protocol and prevent signaling along a circuit using dummy messages. Sending such messages will throw off the counter by some routers on the circuit and the circuit will stop working. We can model a stream cipher by expressing the encryption of the ith message p with key k as $\{p\}_{(k,i)}$, and allowing a different permutation to be applied for every pair (k, i). This can only increase the size of the configuration indistinguishability relation. However, the proof for the distinguishability of configurations only relies on the ability of the adversary to decrypt using his keys, count messages, and recognize the circuit identifier. Therefore it still holds when the model uses a stream cipher. Also, with a stream cipher the circuit identifier is still not necessary for our results. The adversary can again use the process described above to link entries in compromised routers, since although it involves sending dummy messages, they are sent after the circuit creation is finished and therefore do not interfere with it.

Allowing users to create multiple circuits doesn't weaken the adversary's power to link together its circuit positions and determine their position, but the number of configurations that are consistent with this view does in some cases increase. Let users create an arbitrary number of circuits. The adversary can still link positions and count messages as before, so the adversary can distinguish configurations C and D if $\rho(C) \neq \rho(D)$. However, Proposition 1 does not continue to hold, as it is no longer necessary for there to be more than just user u with an uncompromised first router to prevent u from being identified with its circuit. It can, however, be shown that the converse of Theorem 4 continues to hold if we replace "$C \approx_{D_A} D$" with "$\rho(C) = \rho(D)$."

5 Conclusions

We have presented a model of onion routing and characterized when anonymity and unlinkability are provided. The model uses IO automata and provides asynchronous communication. The onion routing protocol we describe is based on the Tor protocol and is connection-oriented. The adversary we analyze is local and active in the sense that he is allowed to run arbitrary automata but is limited to the view of a subset of users and routers that he controls. We show that the adversary can determine when his routers hold positions in the same circuit and where in the circuit they are located, and only this. This gives a simple set of conditions for sender anonymity, receiver anonymity, and unlinkability, that basically just require that the first or last router in a circuit is uncompromised.

Two directions for future work on modeling onion routing are improving the model and improving the analysis. A big missing piece in the current model is the lack of time. Timing attacks are successful in practice, and we have attempted to include one attack of this sort in the model by using circuit identifiers, but this is just an approximation. Also, we have simplified the Tor protocol by omitting key exchange, circuit teardowns, the final unencrypted message forward, and stream management and congestion control. Adding some or all of these would bring the model closer to reality. Towards improving the analysis, we have made several assumptions about the cryptosystem without exhibiting an encryption scheme for which they hold, and this should be done. Also probabilities in both the behavior of the users and the operation of system should be added to the model and analyzed according to probabilistic definitions of anonymity.

References

[1] Adam Back, Ulf Möller, Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. *Information Hiding (IH 2001)*, pp. 245-257. Springer-Verlag, LNCS 2137, 2001.

[2] Philippe Boucher, Adam Shostack, and Ian Goldberg. "Freedom Systems 2.0 Architecture." *Zero Knowledge Systems, Inc. White Paper*, 2000.

[3] Jan Camenisch and Anna Lysyanskaya. "A Formal Treatment of Onion Routing." *CRYPTO 2005*, pp. 169.187, 2005.

[4] David Chaum. "The dining cryptographers problem: Unconditional sender and recipient untraceability." *Journal of Cryptology: The Journal of the International Association for Cryptologic Research*, 1(1), pp. 65-75, 1988.

[5] David Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms." *Communications of the ACM*, 24(2), pp. 84-88, 1981.

[6] George Danezis, Roger Dingledine, and Nick Mathewson. "Mixminion: Design of a type III anonymous remailer protocol." *IEEE Symposium on Security and Privacy*, pp. 2-15, IEEE CS, 2003.

[7] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. "Towards measuring anonymity." *Privacy Enhancing Technologies Workshop 2002*, 2002.

[8] Roger Dingledine, Nick Mathewson, and Paul Syverson. "Challenges in deploying low-latency anonymity (DRAFT)." Available at *<http://tor.eff.org/cvs/tor/doc/design-paper/challenges.pdf>*, 2005.

[9] Roger Dingledine, Nick Mathewson, and Paul Syverson. "Tor: The Second-Generation Onion Router." *13th USENIX Security Symposium*, 2004.

[10] Michael Freedman and Robert Morris. "Tarzan: A peer-to-peer anonymizing network layer." *ACM Conference on Computer and Communications Security*, pp. 193-206, 2002.

[11] David Goldschlag, Michael Reed, and Paul Syverson. "Hiding routing information." *Proceedings of the First International Workshop on Information Hiding*, pp. 137-150, Springer-Verlag, LNCS 1174, 1996.

[12] Joseph Halpern and Kevin O'Neill. "Anonymity and Information Hiding in Multiagent Systems." *Workshop on Formal Methods in Security Engineering*, pp. 63-72, 2005.

[13] Andrew Hintz. "Fingerprinting websites using traffic analysis." *Privacy Enhancing Technologies*, pp. 171-178, Springer-Verlag, LNCS 2482, 2002.

[14] Nancy Lynch. "Distributed Algorithms." Morgan Kaufmann, San Francisco, CA, first edition, 1996.

[15] Sjouke Mauw, Jan Verschuren, and Erik de Vink. "A Formalization of Anonymity and Onion Routing." *European Symposium on Research in Computer Security (ESORICS)*, pages 109-124, 2004.

[16] Steven J. Murdoch and George Danezis. "Low-Cost Traffic Analysis of Tor." *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, May 2005.

[17] Michael Reed, Paul Syverson, David Goldschlag. "Anonymous connections and onion routing." *IEEE Journal on Selected Areas in Communications*, 16(4): 482-494, 1998.

[18] Michael Reiter and Aviel Rubin. "Crowds: anonymity for Web Transactions." *ACM Transaction on Information and System Security* 1(1), pp. 66-92, 1998.

[19] Andrei Serjantov and George Danezis. "Towards an Information Theoretic Metric for Anonymity." *Privacy Enhancing Technologies Workshop*, 2002.

[20] Paul Syverson, Michael Reed, and David Goldschlag. "Onion Routing access configurations." *DARPA Information Survivability Conference and Exposition (DISCEX 2000)*, 1, pp. 34-40, IEEE CS Press, 2000.

[21] Paul Syverson, Gene Tsudik, Michael Reed and Carl Landwehr. "Towards an Analysis of Onion Routing Security." *Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pp. 96-114, 2000.